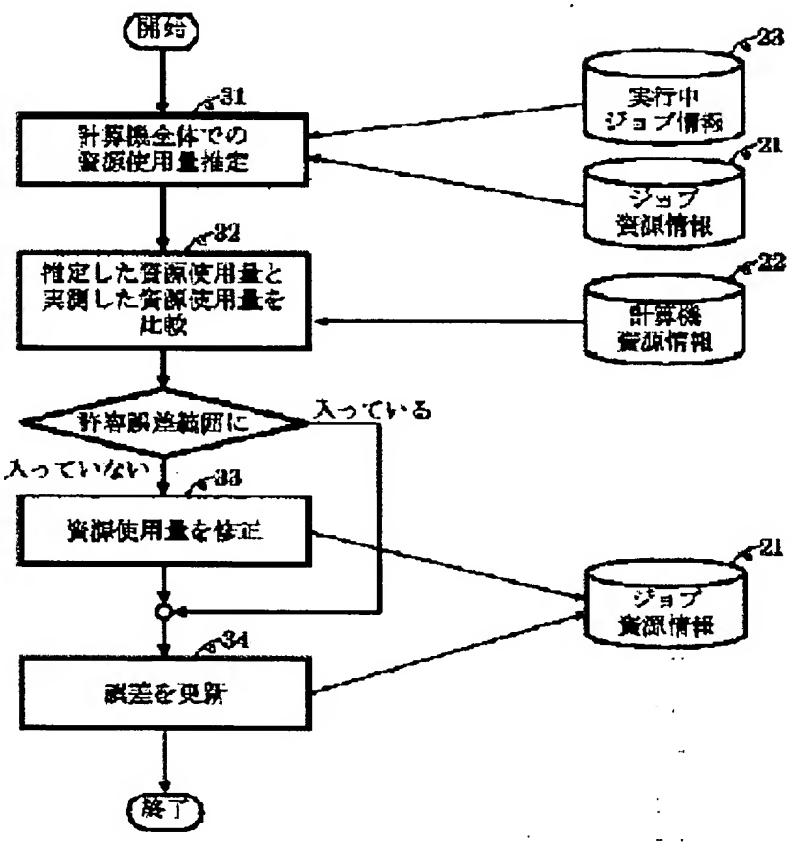
# MicroPatent® PatSearch FullText: Record 1 of 1

Search scope: US Granted JP (bibliographic data only)
Years: 1991–2003
Patent/Publication No.: JP09311795




Order This Patent | Family Lookup | Find Similar | Legal Status

Go to first matching text

JP09311795 A
SCHEDULING METHOD
HITACHI LTD
Inventor(s): ;KURODA SAWAKI ;FUKUZAWA JUNJI ;YAMAMURA NOBUO ;MORITA SHINJI
Application No. 08126847 JP08126847 JP, Filed 19960522,A1 Published 19971202

Abstract: PROBLEM TO BE SOLVED: To automatically acquire a system resource quantity needed to execute individual jobs by updating the difference between a measured value of the resource consumption of a computer and an estimated value and error in the resource demand quantity of a job being executed.

SOLUTION: At each resource, job information 32 in execution and job resource information 21 are totalized to find the estimated value of the resource consumption of the whole computer (step 31), and the found estimated value of the resource consumption is compared with the measured value of the consumption of the resource information 22

of the computer (step 32). When an error in the resource consumption of the whole computer is exceeded, the resource consumption is corrected according to the rate of the resource demand quantity and error (step 33), and the job resource information in execution and further the difference between the measured value of the resource consumption of the computer and the estimated value of the resource consumption and the error in the demand quantity of the job resource information 21 in execution are updated (step 34). Therefore, the system resource quantity needed to execute individual jobs can automatically be acquired.

Int'l Class: G06F00946; G06F01516

Patents Citing this One: No US, EP, or WO patents/search reports have cited this patent.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平9－311795

(43)公開日　平成９年(1997)12月２日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | ＦＩ | | 技術表示箇所 |
|---|---|---|---|---|---|
| Ｇ０６Ｆ　9/46 | ３４０ | | Ｇ０６Ｆ　9/46 | ３４０Ｆ | |
| | | | | ３４０Ｄ | |
| 15/16 | | | 15/16 | ４２０Ｊ | |

審査請求　未請求　請求項の数６　ＯＬ　（全９頁）

(21)出願番号　特願平8－126847

(22)出願日　平成８年(1996)５月22日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　黒田　沢希
神奈川県川崎市麻生区王禅寺1099番地株式会社日立製作所システム開発研究所内

(72)発明者　福澤　淳二
神奈川県川崎市麻生区王禅寺1099番地株式会社日立製作所システム開発研究所内

(72)発明者　山村　宣夫
神奈川県横浜市戸塚区戸塚町5030番地株式会社日立製作所ソフトウェア開発本部内

(74)代理人　弁理士　小川　勝男

最終頁に続く

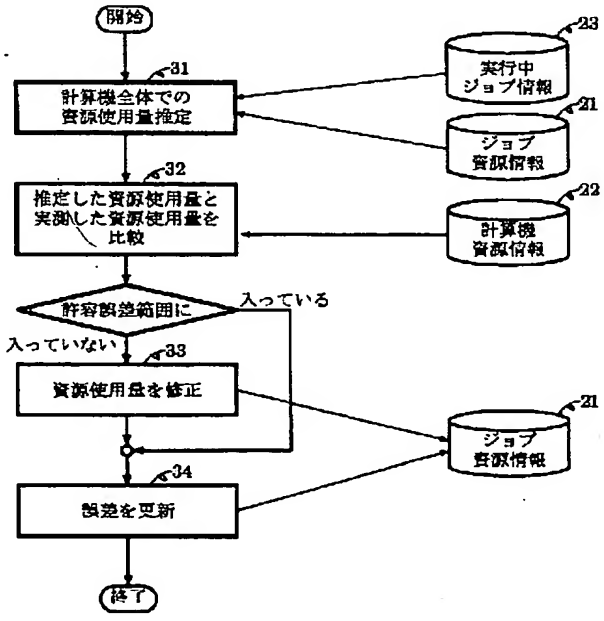
(54)【発明の名称】　スケジュール方法

(57)【要約】

【課題】ジョブ投入後のシステムの状態を予測するために、ジョブの特性を自動的に収集することを目的とする。

【解決手段】分散計算機システムにおいて、個別のジョブの使用する資源量の情報とその確度を備え、実行中のジョブの前記資源量の合計と、計算機システムの負荷の実測値との差を、使用資源量情報確度に応じてジョブの使用する資源量の情報を修正することによって、ジョブの使用する資源量の情報を獲得し、ジョブの使用する資源量を元に、特定の資源に集中しないスケジューリングを行なう。

図３



【特許請求の範囲】

【請求項１】計算機システムが提供する資源と、計算機システム全体での資源の使用量を測定する手段と、該計算機上で実行中のジョブと、実行中ジョブの一覧情報と、を備える計算機システムにおける、ジョブ資源要求量獲得方法であって、

ジョブの名称と、計算機の各資源について当該ジョブが要求する量の推測値と、その確からしさをあらわす確度とを含むジョブ資源情報を備え、

(a) 計算機で実行中のジョブについて、ジョブ資源情報の当該ジョブが要求する資源量を合計して計算機全体の資源使用量の推測値をもとめ、当該ジョブに関する資源要求量の誤差を加算して、計算機全体の資源使用量の誤差をもとめ、

(b) 計算機の資源使用量の実測値と、前記求めた資源使用量の推測値との差が、前記計算機全体の資源使用量の誤差より大きい時、

(c) 資源使用量の実測値と推測値の差がなくなるように、当該計算機で実行中のジョブについて、資源要求量およびその誤差の割合に応じて、実行中のジョブ管理情報の推定資源要求量に加え、

(d) 前記ステップ(b) で求めた実測値と推測値の差と、実行中のジョブの資源要求量の誤差を更新することを特徴とした、スケジュール方法。

【請求項２】請求項１の計算機システムで、さらに計算機の持つ資源容量の情報を備え、前記ステップ(a) において、

(a1) 実行中のジョブおよび各資源について、当該ジョブの資源推定要求量の合計を出力し、

(a2) 資源推定要求量合計と、計算機の資源容量と、各ジョブの推定資源要求量とからジョブの稼働率を算出し、

(a3) 各ジョブの稼働率から、ジョブの資源推定使用量と、処理時間の推定値を出力することを特徴とした、スケジュール方法。

【請求項３】請求項１の計算機システムにおいて、スケジューラがジョブの起動／終了を検知するステップを備え、

前記検知した時に、前記各ステップを実行することを特徴とする、スケジュール方法。

【請求項４】分散計算機環境で、ジョブを投入する時、

(e) 請求項１で求めたジョブの資源使用量を元に、当該ジョブの資源推定使用量情報を求め、

(f) 当該ジョブの資源推定使用量を、各計算機の資源使用量に加えて、当該ジョブ投入後の資源使用量を求め、

(g) 資源使用量と資源容量の比を、各資源について和を求め、

(h) 前記和がもっとも小さくなる計算機に当該ジョブを割り当てることを特徴とする、スケジュール方法。

【請求項５】請求項１のステップ(f) の代わりに、

(f') 実行中のジョブジョブと、投入するジョブについて、資源推定使用量を読み出し、前記実行中および投入ジョブの資源推定使用量の総和を求め、これを資源使用量とすることを特徴とした、スケジュール方法。

【請求項６】請求項１のステップ(f) の代わりに、

(f") 当該ジョブの資源推定使用量と誤差範囲の割合との積を、各計算機の資源使用量に加えて、当該ジョブ投入後の資源使用量を求める、ことを特徴とした、スケジュール方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は繰り返し実行されるジョブに関して、実行時のシステムに与えるCPU負荷やI0 負荷等から当該ジョブが必要とする負荷を推定し、当該推定値を使って、ジョブ投入後のシステムの状態を予測して、ジョブを投入する計算機を決定するスケジューラに関する。

【０００２】

【従来の技術】"PVM Guide" (http://csepl.phy.ornl.gov/pvm_guide/pvm_guide.html)によると、分散計算機環境で性能向上を図る手段として、負荷分散がある。各計算機の実行待ちジョブ数を計測し、この実行待ちジョブ数の最も少ない計算機で新しいジョブを実行することによって、各計算機の負荷を平均化することができる。

【０００３】また、Greoge A. Champine 著の『Distributed Computer Systems:Impact onManagement, Design, and Analysis』によると、個別ジョブがシステムに与える負荷を得る方法として、ジョブの実行ステップ数等をエミュレータ等を利用して計測する方法がある。この方法では、ジョブの動作が詳細にわかるため、資源に与える負荷の質／量／時間をほぼ正確に取得することができる。しかし、１命令をエミュレータで解析／実行するので、ジョブの実行ステップ数の計測には非常に時間がかかる。またジョブが他社の開発した物の場合、実行ステップ数の計測は、使用契約上認められないことがある。

【０００４】個別ジョブがシステムに与える負荷を得る別の方法として、計算機上でそのジョブだけを動かして、計算機の負荷を実測する方法がある。この方法では、短時間で負荷値を測定できる。しかし、目的のジョブ以外の影響を排除するために、測定専用に計算機システムを用意する必要がある。

【０００５】

【発明が解決しようとする課題】前記システムの負荷状態による負荷分散スケジュールでは、長時間かかるジョブを同一計算機に割り当てる場合があり、その結果全体の処理時間が伸びる場合があるという問題がある。

【０００６】個別ジョブの実行負荷を得るために、当該ジョブの実行ステップ数等を計測する方法では、ジョブの実行ステップ数の計測は通常シミュレータを利用する

ため、非常に時間がかかるという問題がある。またジョブが他社の開発した物の場合、実行ステップ数の計測は、その使用契約上認められないことがあるという問題もある。

【０００７】個別ジョブの実行負荷を得るために、計算機上で当該ジョブだけを動かして負荷を実測する方法では、測定専用に分散システムを用意する必要があり、コストがかかってしまうという問題がある。

【０００８】本発明の目的は、個別ジョブに関して、当該ジョブの実行に必要なシステム資源量を自動的に獲得することと、前記資源量を元にジョブを計算機に割り当てることである。

【０００９】

【課題を解決するための手段】ジョブ名称に対応して、当該ジョブが使用するCPU やIO の負荷およびジョブ所要時間などの推定値と、その確度を備えた情報を備える。

【００１０】分散した計算機毎にCPU ／IO 能力などの値を測定する。

【００１１】未知のジョブの場合、負荷の推定値および確度は、あらかじめ定めてあるデフォルト値を設定する。

【００１２】ジョブ実行後の計算機の負荷変動を、そのジョブの推定負荷として登録する。

【００１３】ジョブの実行中に実行中のジョブの負荷推定値の合計と、実測した計算機の負荷との差が、確度から求められる許容誤差の範囲内であれば、確度を上げる。

【００１４】ジョブの実行中に、実行中のジョブの負荷推定値の合計と、実測した計算機の負荷との差が、確度から求められる許容誤差範囲を越えた時、推定の誤差の割合に対応して、推定の負荷の大きさに比例して、その差分を分配して負荷推定値を修正し、確度を下げる。

【００１５】これらのステップにより、個別ジョブが計算機システムに与える負荷を自動的に取得する。

【００１６】

【発明の実施の形態】以下に本発明の一実施の形態を説明する。

【００１７】図１は本発明によるスケジュール方式を適用した分散計算機システム(1) の概要を示す。この分散計算機システム(1) は、管理計算機(11) と、複数の計算サーバ(12) と、それら繋ぐネットワーク(13) とを含む。なお、管理計算機(11)は計算サーバ(12) の一つを兼用しても構わない。

【００１８】管理計算機(11) は、ジョブ資源情報(21) と、当該計算機システムに要求された起動待ちのジョブ(112) を管理するジョブ受付キュー(111) と、当該ジョブを１つの計算サーバ(12) に割り当てるスケジューラ(113) を備える。

【００１９】計算サーバ(12) は、CPU 時間、メモリ容量、ファイルアクセス頻度、ネットワークアクセス頻度など、複数の資源(121) と、資源の容量や使用量を保持する計算機資源情報(22) と、資源を使用するジョブ(123) と、ジョブを管理する実行中ジョブ情報(23) と、計算サーバ(12) の資源(121) の状態を調べる資源使用量測定手段(122) と、ジョブ情報獲得手段(124) を備える。

【００２０】ジョブ情報獲得手段(124) は、資源使用量測定手段(122) から計算サーバ(12)の資源の使用量と、実行中ジョブ情報(23) から、実行中のジョブ(123) が使用する資源(121) を推測し、ジョブが使用する資源の量を記憶するジョブ資源情報(12)を更新する。

【００２１】ジョブの実行要求があると、実行要求受付キュー(111) に実行待ちジョブ(112)を入れる。スケジューラ(113) は実行要求受付キュー(111) から実行待ちジョブ(112) を取りだし、ジョブ資源情報(21) からジョブが使用する資源量の情報を取得し、各計算サーバ(12) の資源使用量測定手段(122) から資源の状態を取得し、実行待ちジョブ(112) を適当な計算サーバ(12) に割り当てる。

【００２２】図２は、各テーブルの構造を示している。

【００２３】ジョブ資源情報(21) は、分散計算機システム上に１つあり、ジョブ名(211)と、当該ジョブの実行時に消費する各資源名(212) と、その資源を当該ジョブが使用する資源必要量(213) とその確からしさをあらわす確度(214) とを備える。資源としてジョブの起動から終了までの処理時間、単位時間当たりに使用するCPU 時間、占有するメモリ量、ファイルアクセス頻度、ネットワークアクセス頻度などがある。このテーブルはジョブ情報獲得手段(124) が後述の方法によって追加／変更する。

【００２４】計算機資源情報(22) は、各計算サーバにあり、当該計算機が提供できる各資源について、その資源名(221) と、提供できる能力をあらわす資源容量(222) と、資源の使用が著しく悪くなる使用率をあらわす使用限界(223) と、資源に対する当該計算サーバにおける負荷の測定値である資源使用量(224) を備える。このテーブルの資源使用量(224) 以外の項目は、別の手段によってあらかじめ値が入っているものとする。資源使用量(224) は、資源使用量測定手段(122) によって定期的に更新する。

【００２５】実行中ジョブ情報(23) は、実行中のジョブ(123) について、ジョブ名(231)と、ジョブID(232) を備える。新規起動したジョブについて、当該ジョブのためのエントリを追加し、終了したジョブに関しては、当該終了したジョブのエントリを削除する。

【００２６】以下では、前記の構成を持つ分散計算機システムにおけるジョブ情報獲得方法とスケジュール方法について、説明する。

【００２７】図３はジョブ情報獲得手段におけるジョブ

情報獲得方法の詳細を示す。

【００２８】定期的に以下の処理をおこなう。

【００２９】ステップ31 では、各資源について、実行中ジョブ情報(23) のそれぞれのジョブ名(231) と、ジョブ資源情報(21)のジョブ名(211) と一致するエントリから、から当該ジョブの資源要求量(213)を読みだす。

【００３０】実行中のジョブ計算機上で複数のジョブが動いている時、それぞれの資源に対してジョブは互いに競合しあうため、ジョブの稼働率が下がる。

【００３１】計算機m の資源s の使用量推定値LAms は、実行中のジョブp とその資源要求量推定値LRps と計算機m の資源容量LCms を使って、

【００３２】

【数１】

$$LA_{ms} = \min\left(LC_{ms}, \sum_{p} LR_{ps}\right) \quad \cdots\cdots\cdots \quad (数1)$$

【００３３】とし、計算機m でのジョブp の稼働率Amp は、資源s の容量LCms、ジョブp の資源要求量LRps を使って、

【００３４】

【数２】

$$A_{mp} = \begin{pmatrix} \frac{LR_{ps}}{\sum LR_{ps}} & |LC_{ms} < \sum LR_{ps}\text{の時} \\ 1 & |LC_{ms} > \sum LR_{ps}\text{の時} \end{pmatrix} \cdots\cdots\cdots (数2)$$

【００３５】によって求める。

【００３６】次にステップ32 では、計算機資源情報(22) の実測した資源使用量(224) と、前記使用量の推定値LAms との差を求める。後の説明のためにこの差を実推差と呼ぶ。また、資源要求量(213) と確度(214) から求める許容誤差を求める。

【００３７】許容誤差Eps は、ジョブp の必要資源量LRps と、その確度Rps と、を用いて、

【００３８】

【数３】

$$E_{ps} = (1 - R_{ps}) \cdot LR_{ps} \quad \cdots\cdots\cdots \quad (数3)$$

【００３９】とする。

【００４０】前記許容誤差より実推差が大きい時、ステップ33 では、ジョブの実測値(224)と推定使用量LAms とが一致するように、実推差を実行中のジョブの資源必要量に分配する。このとき、ジョブの資源要求量が大きいものがよりたくさん、確度の低いものがよりたくさん、変化するようにする。

【００４１】ジョブp の必要資源量LRps を、その確度Rps と、計算機m での測定値LSms、推定値LAms、定数c を用いて、

【００４２】

【数４】

$$LR_{ps} \Leftarrow LR_{ps} + c \cdot A_{mp} \cdot \frac{LR_{ps} \cdot (1 - R_{ps})}{\sum_p (LR_{ps} \cdot (1 - R_{ps}))} \cdot (LS_{ms} - LA_{ms}) \quad \cdots\cdots\cdots (数4)$$

【００４３】で更新する。

【００４４】ステップ34 では、実推差と、ジョブの資源必要量(213) とその確度(214) から、ジョブの資源必要量の確度を変更する。

【００４５】ジョブp の資源s に関する資源必要量の確度Rps を、計算機m での資源使用量測定値LSms と、資源使用量推定値LAms と、許容誤差Eps と、定数c を用いて、

【００４６】

【数５】

$$R_{ps} \Leftarrow R_{ps} + c \cdot (E_{ps} - |LS_{ms} - LA_{ms}|) \cdot A_{mp} \cdot \frac{|LS_{ms} - LA_{ms}|}{LA_{ps}} \quad \cdots\cdots\cdots (数5)$$

【００４７】で更新する。

【００４８】以上の処理を各計算サーバ(12) の各資源について繰り返す。

【００４９】次に図４はスケジューラのジョブ起動割当手段の詳細を示す。

【００５０】ジョブ実行要求キュー(111) に実行待ちのジョブがある時、以下の処理をおこなう。

【００５１】ステップ41 では、実行予定ジョブをジョブ実行要求キュー(111) から取得し、ジョブ資源情報(21) からジョブ名(211) が合致するエントリの各資源について、資源必要量(213) とその確度(214) を取得する。

【００５２】ステップ42 では、計算機資源情報(22) から、実測した資源使用量(224) を取得し、ステップ41

で得たジョブの資源必要量(213) とその確度(214) を利用して、ジョブ投入後の資源使用量を推定する。

【００５３】ステップ43 では、各計算機の資源容量(222) と、前記ジョブ投入後の資源使用量推定値とから、推定される資源使用率を求め、これを使用限界(223) と比較する。

【００５４】推定される資源使用率が使用限界(223) 以下となる計算サーバ(12) がある時、ステップ44 では、

$$0 > \min_{m}\left(\max_{s}\left(\frac{LS_{ms}+LR_{ps}}{LC_{ms}}-B_{ms}\right)\right)\cdots\cdots\cdots(数6)$$

【００５７】の条件を満たす。

【００５８】前記条件を満たす計算サーバ(12) がない時、ステップ45 では、当該ジョブを起動しないで、実行待ちキュー(111) に保留し、一定時間後に再びステップ４１から処理する。

【００５９】別のスケジューリング方法として、前記ステップ４２　において、実行中ジョブ情報(23) から、実行中のジョブの一覧を取得し、そのそれぞれの資源必要量を求め、ステップ41 で得たジョブの資源必要量(213) と合計することによってジョブ投入後の資源必要量を推定する。

【００６０】すなわち、ステップ44 で選択される計算機m は、ジョブp 投入時に、資源sについて、投入前の実測値LSms と、投入するジョブの資源要求量LRps と、資源容量LCms と、資源使用限界Bms とにおいて、

【００６１】

【数７】

$$0 > \min_{m}\left(\max_{s}\left(\frac{\sum_{p=実行中}(LR_{ps})+LR_{ps}}{LC_{ms}}-B_{ms}\right)\right)\cdots\cdots\cdots(数7)$$

【００６２】の条件を満たす。

【００６３】別のスケジューリング方法として、前記ステップ44 において、全てのマシンの中で、ジョブ投入後の資源使用率の合計が最も低くなる計算機に割り当てる。

【００６４】すなわち、この計算機m は、ジョブp 投入時に、資源s について、投入前の実測値LSms と、投入するジョブの資源要求量LRps と、資源容量LCms と、資源使用限界Bms とにおいて、

【００６５】

【数８】

$$0 > \min_{m}\left(\sum_{s}\left(\frac{LS_{ms}+LR_{ps}}{LC_{ms}}-B_{ms}\right)\right)\cdots\cdots\cdots(数8)$$

【００６６】の条件を満たす。

【００６７】また別のスケジューリング方法として、前記ステップ44 において、全てのマシンの中で、ジョブ投入後の最大の資源使用率の最も低くなる計算機に割り当てる。

【００６８】すなわち、この計算機m は、ジョブp 投入時に、資源s について、投入前の実測値LSms と、投入するジョブの資源要求量LRps と、資源容量LCms と、資源使用限界Bms と、において、

【００６９】

【数９】

$$\min_{m}\left(\max_{s}\left(\frac{LS_{ms}+LR_{ps}}{LC_{ms}}\right)\right)\cdots\cdots\cdots(数9)$$

【００７０】の条件を満たす。

【００７１】ジョブが必要とする資源量が不確かな間は、予測の影響は小さくなり、現状の計算機の負荷だけでスケジュールするのと変わらない。

資源使用率が最も低い計算サーバに当該ジョブを割当てる。

【００５５】すなわち、この計算機m は、ジョブp 投入時に、資源s について、投入前の実測値LSms と、投入するジョブの資源要求量LRps と、資源容量LCms と、資源使用限界Bms とにおいて、

【００５６】

【数６】

【００７２】繰り返し実行されるジョブが必要とするシステム資源量を自動的に獲得する。ジョブが必要とする資源量がわかると、投入後の計算機の資源使用が集中しないようにスケジュールできる。例えば、CPU 負荷をかけるがIO 負荷をかけないジョブと、CPU 負荷をかけないがIO 負荷をかけるジョブと、を組み合わせてスケジュールするようになり、計算機上のジョブ間の資源の奪い合いが少なくなり、効率的に計算機を使用できる。

【００７３】

【発明の効果】個別のジョブが必要とする複数のシステム資源の量を自動的に獲得できる。この情報によって、特定のシステム資源に処理が集中することによって処理効率が落ちることを防ぐことができる。

【００７４】個別ジョブの処理時間を推定できる。

【図面の簡単な説明】

【図１】本発明の実施例に係わる分散計算機システムの構成図である。

【図２】テーブルのテーブルの構成図である。

【図３】個別ジョブの資源要求量獲得方法のブロック図である。

【図４】ジョブ起動計算機割当スケジュール方法のブロック図である。

【符号の説明】

LRps …ジョブp が単位時間に必要な資源s の量の推定値、

LRms …計算機m 上のジョブが単位時間に必要な資源s の量の推定値、

LAps …ジョブp が単位時間に使用する資源s の量の推定値、

LAms …計算機m が単位時間に使用する資源s の量の推定値、

LSms …計算機m が単位時間に使用した資源s の量の実測値、

LCm …計算機m の単位時間に使用可能な資源s の容量、

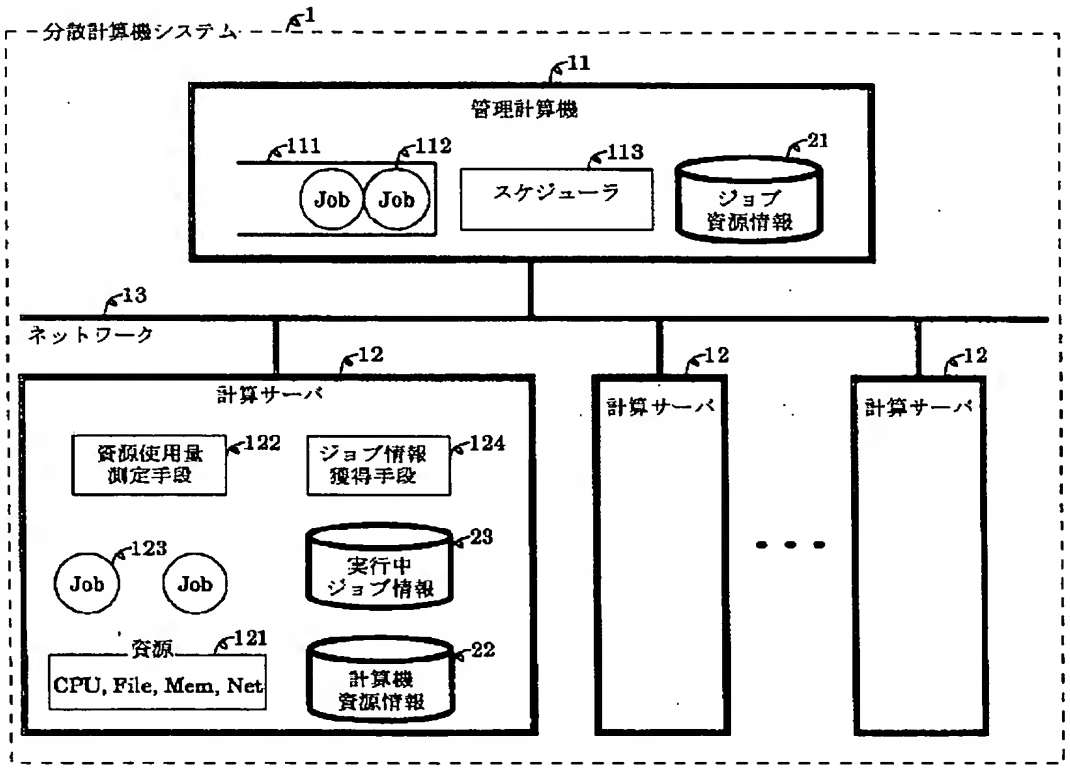
Eps …LAps の誤差、　　ERps …LAps の誤差率、＝Eps/LAps[%]、

Rps …LAps の確度、＝(LAps-Eps)/LAps、

Amp …計算機m 上のジョブp の稼働の割合、Bms …資源使用限界。

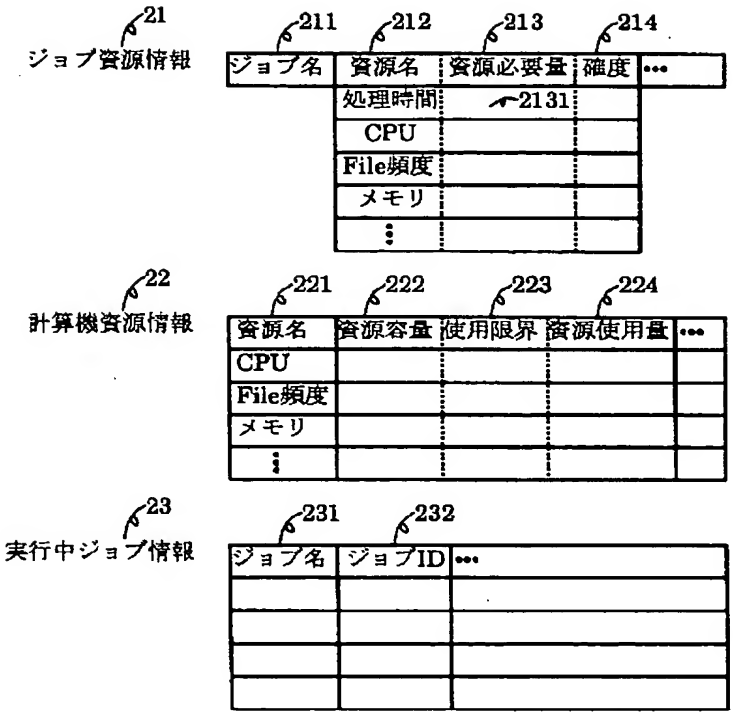
【図１】

図1



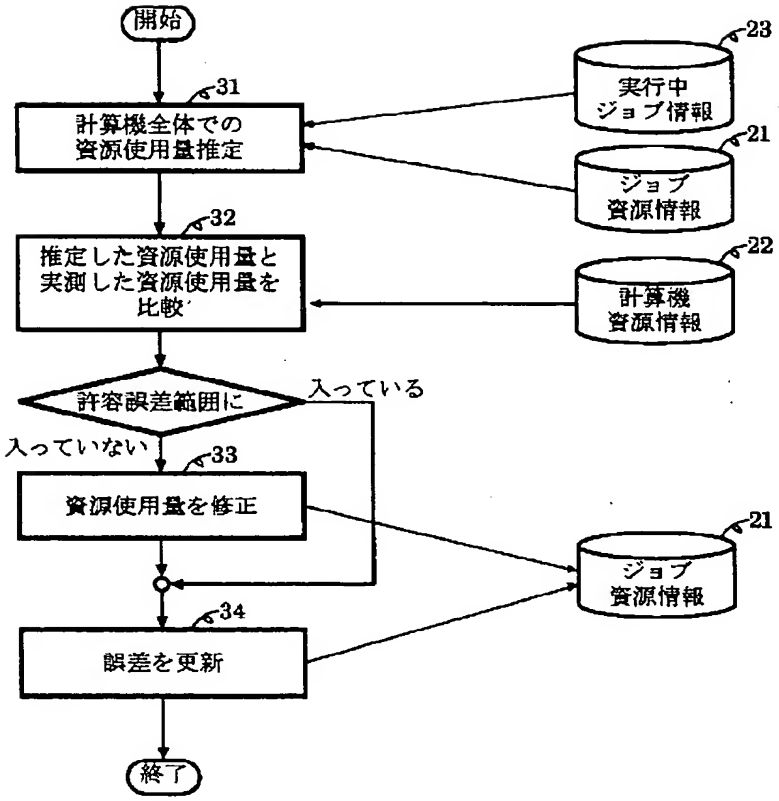
【図２】

図2


21
ジョブ資源情報
211
212
213
214
ジョブ名
資源名
資源必要量
確度
処理時間
2131
CPU
File頻度
メモリ
22
計算機資源情報
221
222
223
224
資源名
資源容量
使用限界
資源使用量
CPU
File頻度
メモリ
23
実行中ジョブ情報
231
232
ジョブ名
ジョブID

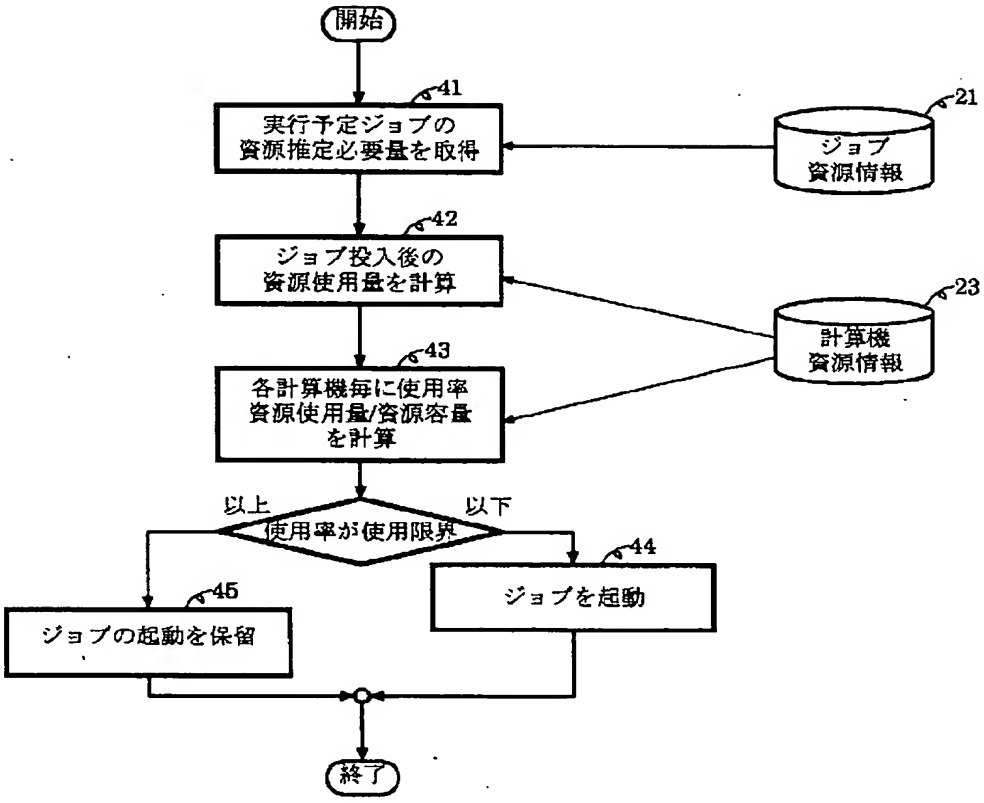
【図３】

図３


開始
31
計算機全体での
資源使用量推定
32
推定した資源使用量と
実測した資源使用量を
比較
許容誤差範囲に
入っている
入っていない
33
資源使用量を修正
34
誤差を更新
終了
23
実行中
ジョブ情報
21
ジョブ
資源情報
22
計算機
資源情報
21
ジョブ
資源情報

【図４】

図４




フロントページの続き

(72)発明者　森田　眞司
神奈川県横浜市戸塚区戸塚町5030番地株式
会社日立製作所ソフトウェア開発本部内